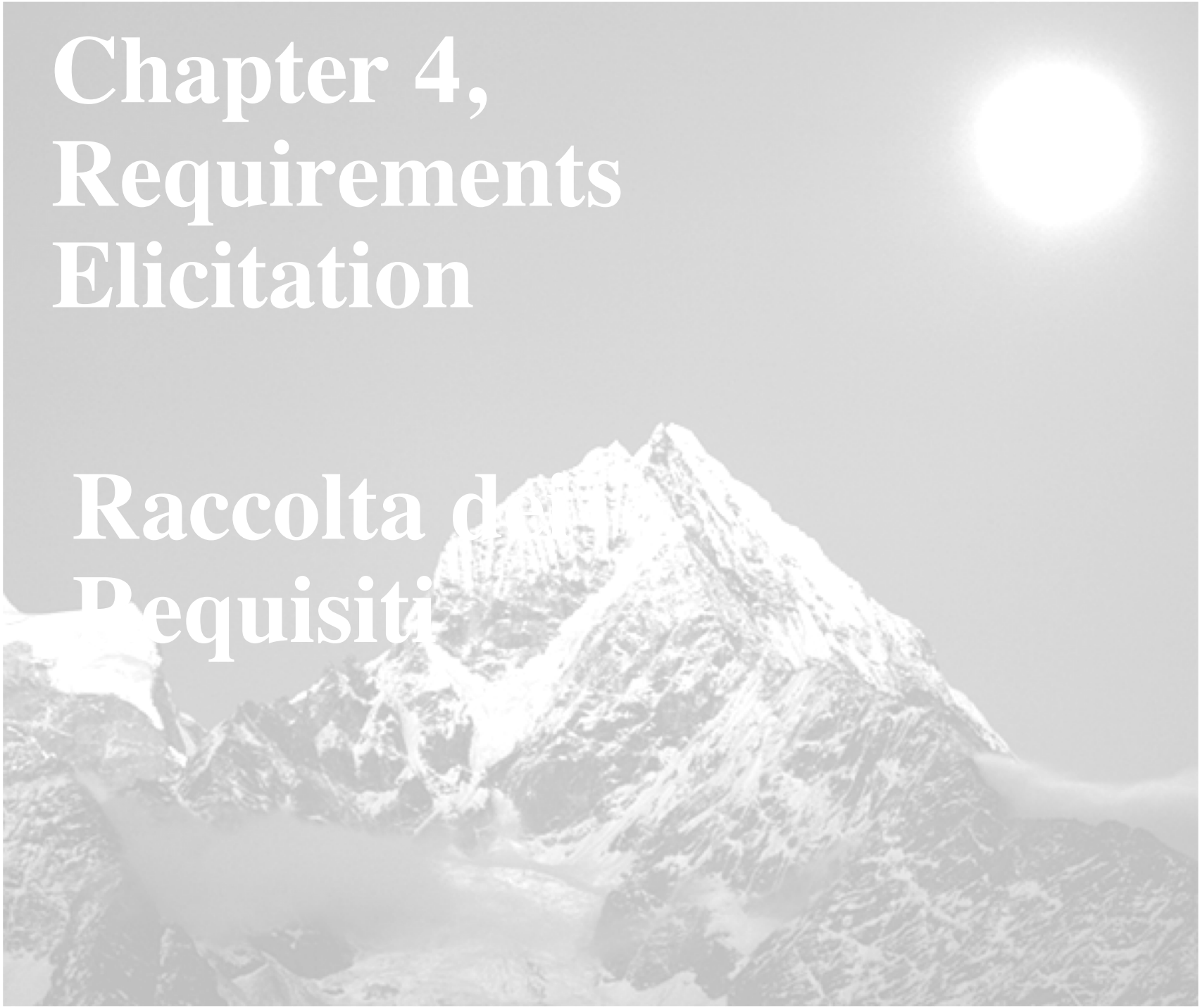
Object-Oriented Software Engineering

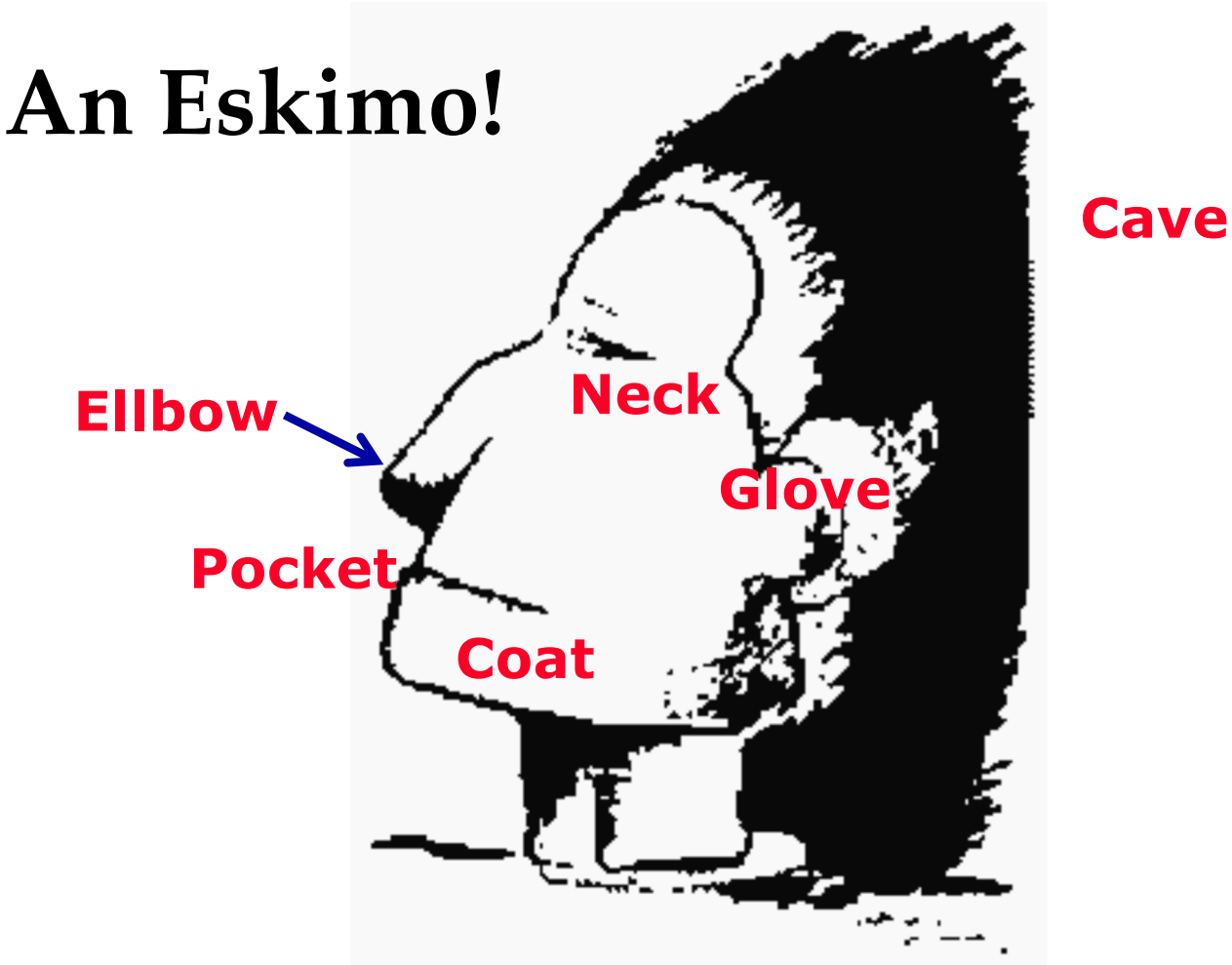
Chapter 4, Requirements Elicitation



Raccolta dei Requisiti

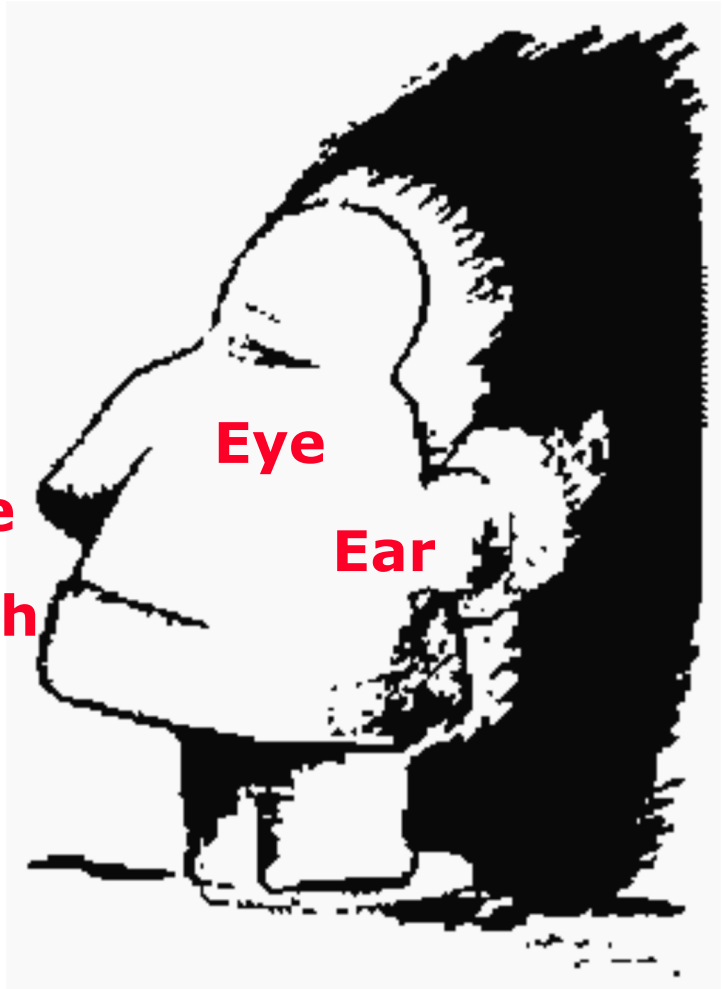
What is this?

What is This?

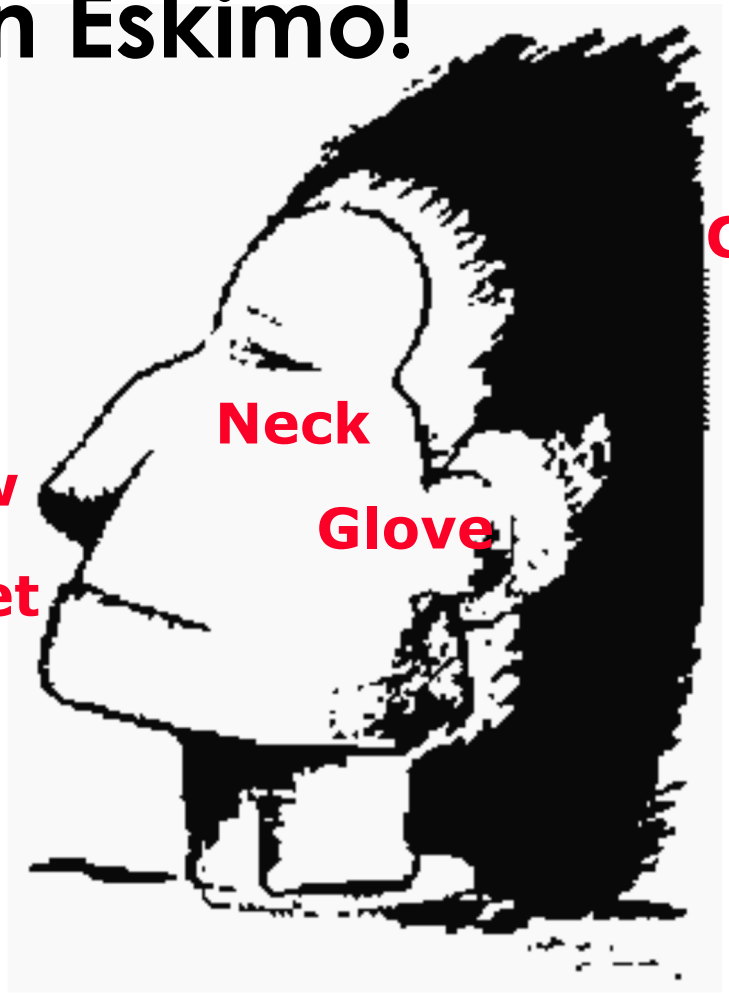


What is This?

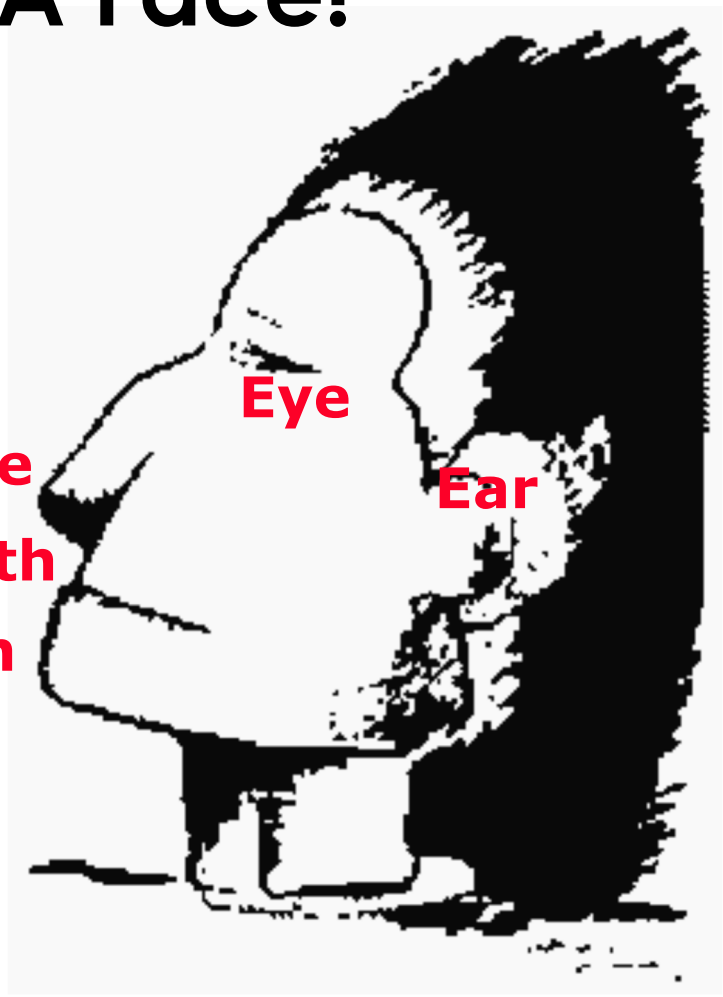
A Face!



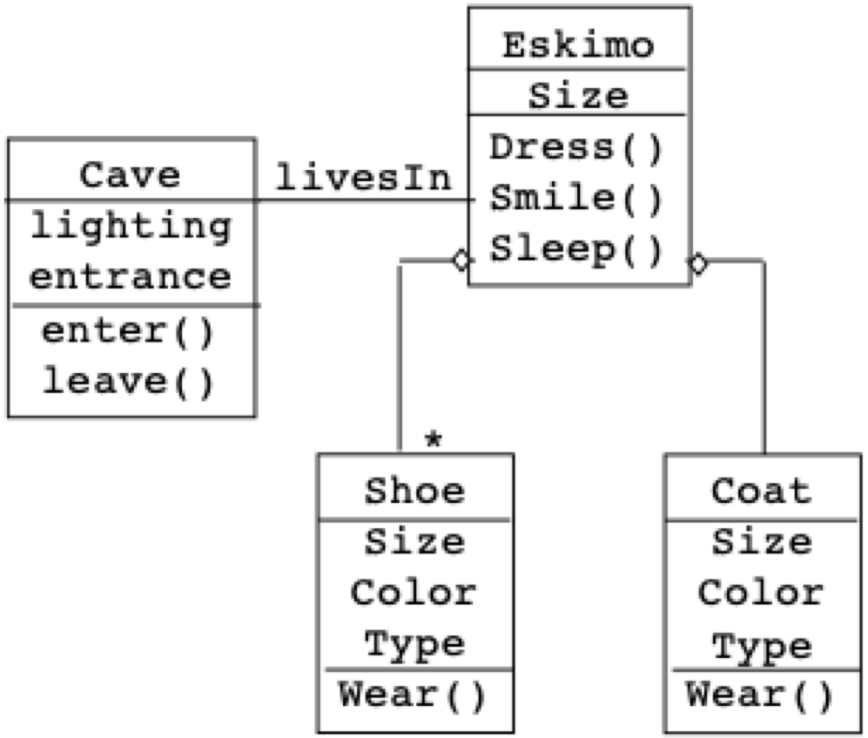
An Eskimo!



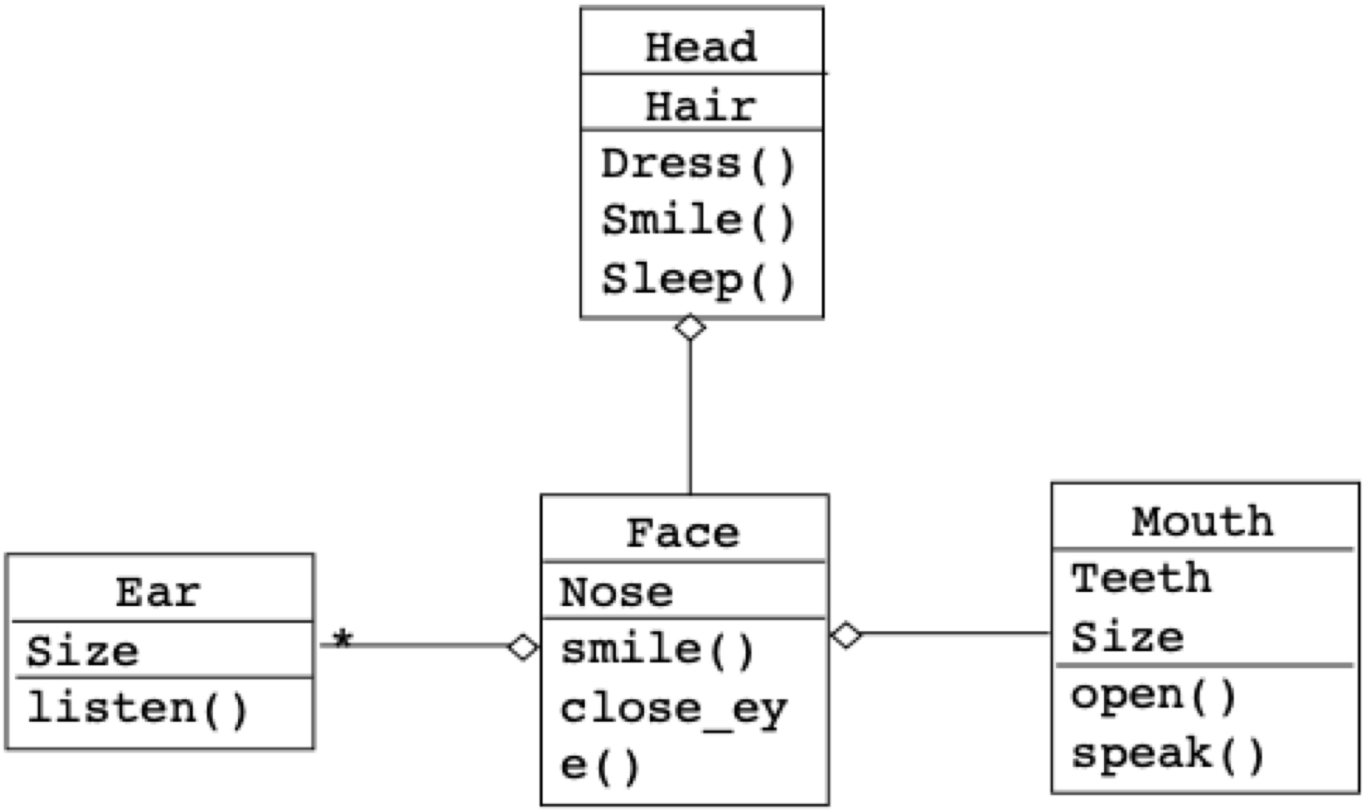
A Face!



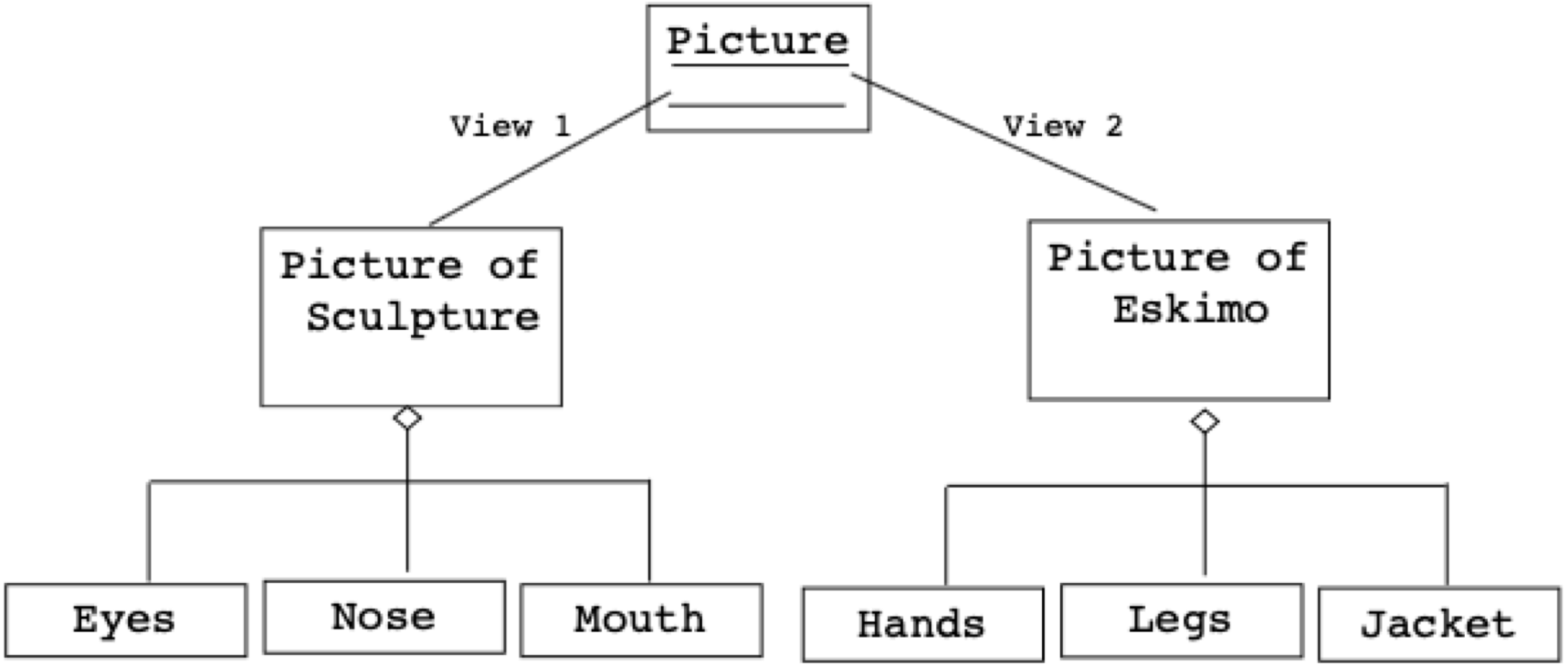
Possible Object Model: Eskimo



Alternative: Head



The Artist's View



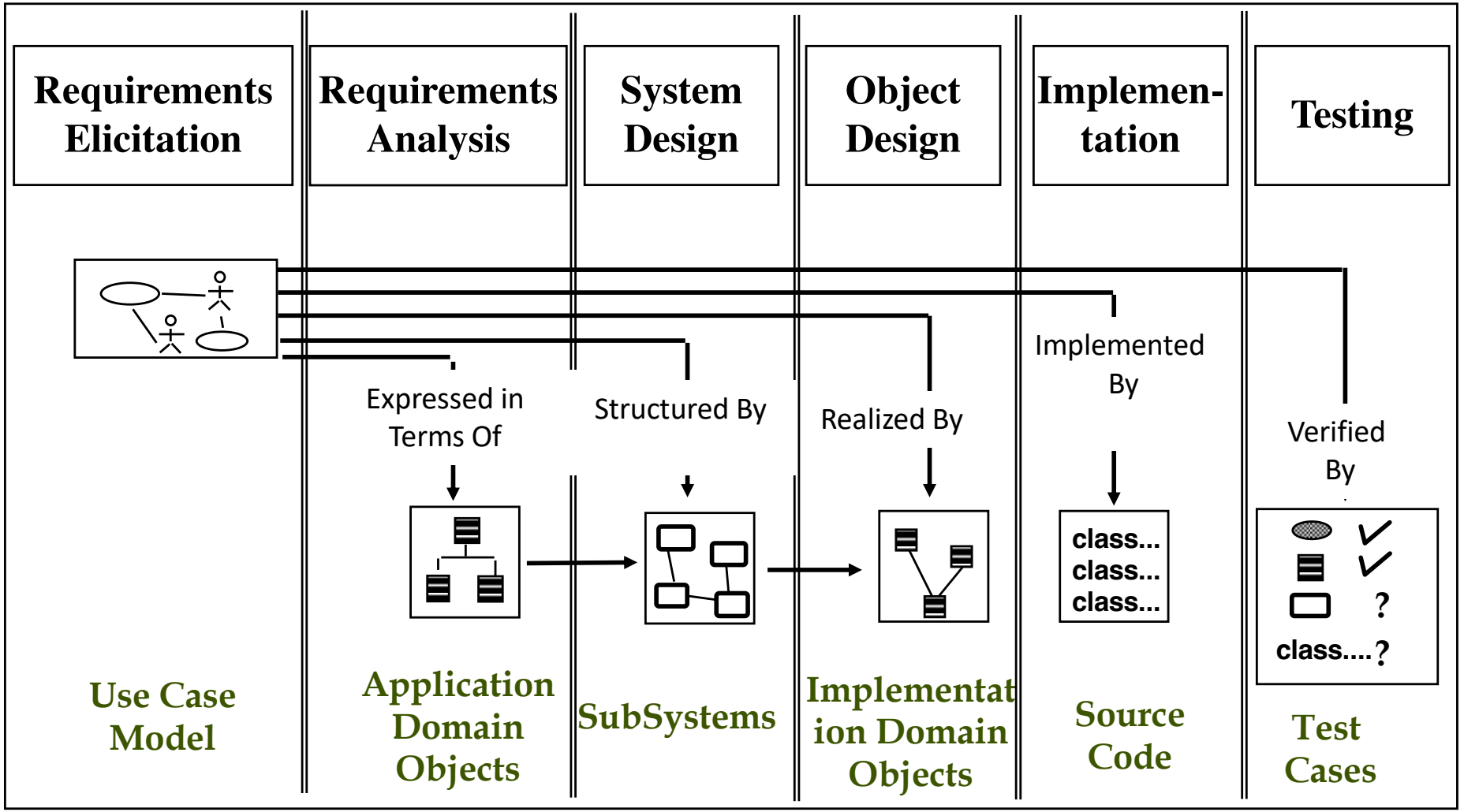
Where are we right now?

- Three ways to deal with complexity:
  - **Abstraction**
  - **Decomposition (Technique: Divide and conquer)**
  - **Hierarchy (Technique: Layering)**
- Two ways to deal with decomposition:
  - **Object-orientation and functional decomposition**
  - **Functional decomposition leads to unmaintainable code**
  - **Depending on the purpose of the system, different objects can be found**
- What is the right way?
  - **Start with a description of the functionality (Use case model). Then proceed by finding objects (object model).**
- What activities and models are needed?
  - **This leads us to the software development lifecycle we use in this class**

Software Development Lifecycle Definition

- Software development lifecycle :
  - **Set of activities and their relationships to each other to support the development of a software system**
- Typical Lifecycle questions:
  - **Which activities should I select for the software project?**
  - **What are the dependencies between activities?**
  - **How should I schedule the activities?**
  - **What is the result of an activity?**

Software Development Lifecycle Activities



First step in identifying the Requirements: System identification

- Two questions need to be answered:
  1. **How can we identify the purpose of a system?**
  2. **(crucial is the boundary)What is inside, what is outside the system?**
- These two questions are answered during requirements elicitation and analysis
- Requirements elicitation:
  - **Definition of the system in terms understood by the customer ("Requirements/System specification")**
- Requirements Analysis:
  - **Definition of the system in terms understood by the developer (Technical specification, "Analysis model")**
- Requirements Process: Contains the activities Requirements Elicitation and Analysis.

Defining the System Boundary is difficult

Example of an Ambiguous Specification

During a laser experiment, a laser beam was directed from earth to a mirror on the Space Shuttle Discovery

The laser beam was supposed to be reflected back towards a mountain top 10,023 feet high

The operator entered the elevation as "10023"

The light beam never hit the mountain top
What was the problem?

The computer interpreted the number in miles...

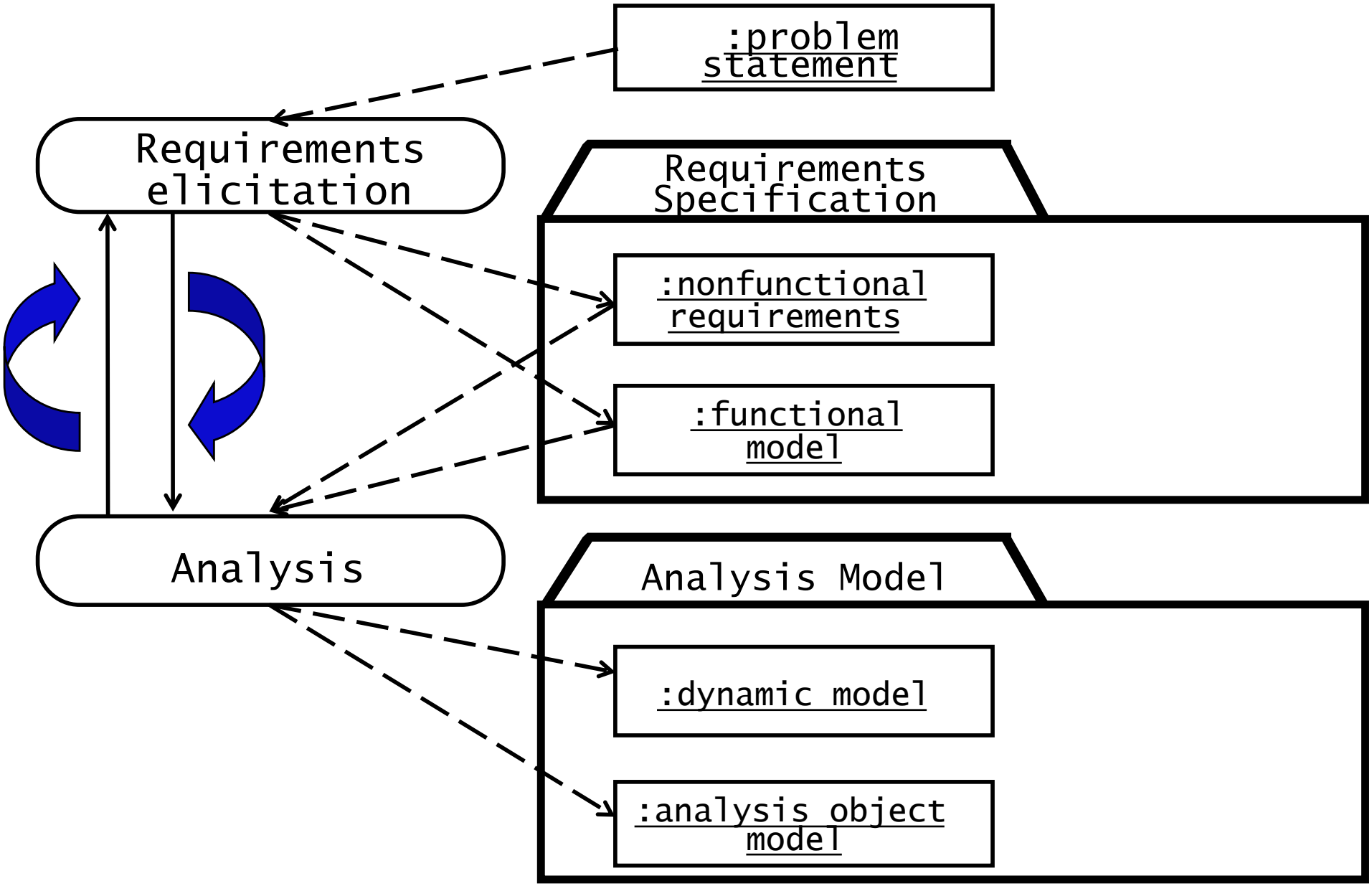
Example of an Unintended Feature

From the News: London underground train leaves station without driver!

What happened?

- A passenger door was stuck and did not close
- The driver left his train to close the passenger door
  - He left the driver door open
  - *He relied on the specification that said the train does not move if at least one door is open*
- When he shut the passenger door, the train left the station without him
  - *The driver door was not treated as a door in the source code*!

Requirements Process



Problem Statement – Statement of Work

- The problem statement is developed by the client as a description of the problem addressed by the system

- A problem statement describes
  - **The current situation**
  - **The objectives**
  - **The functionality the new system should support**
  - **The environment in which the system will be deployed**
  - **Deliverables expected by the client**
  - **Delivery dates**
  - **A set of acceptance criteria.**

Ingredients of a Problem Statement

- Current situation
  - **The problem to be solved**
  - **Description of one or more scenarios**
- Objectives
- Requirements
  - **Functional and nonfunctional requirements**
  - **Constraints (“pseudo requirements”)**
- Target environment
  - **The environment in which the delivered system has to perform a specified set of system tests**
- Project schedule
  - **Major milestones including deadline for delivery**
- Client acceptance criteria
  - **Criteria for the system tests.**

Current situation: The problem to be solved

- There is a problem in the current situation
  - **Examples:**
    - **The response time when playing chess is too slow.**
    - **I want to play Go, but cannot find players on my level.**
- What has changed? Why can address the problem now?
  - ***Change in the application domain***
    - **A new function (business process) is introduced into the business**
  - ***Change in the solution domain***
    - **A new solution (technology enabler) has appeared**

ARENA: The Problem

- The Internet has enabled virtual communities
  - **Groups of people sharing common of interests but who have never met each other in person. Such virtual communities can be short lived (e.g people in a chat room or playing a multi player game) or long lived (e.g., subscribers to a mailing list).**
- Many multi-player computer games now include support for virtual communities.
  - **Players can receive news about game upgrades, new game levels, announce and organize matches, and compare scores.**
- Currently each game company develops such community support in each individual game.
  - **Each company uses a different infrastructure, different concepts, and provides different levels of support.**
- This redundancy and inconsistency leads to problems:
  - **High learning curve for players joining a new community,**
  - **Game companies need to develop the support from scratch**
  - **Advertisers need to contact each individual community separately.**

ARENA: The Objectives

- Provide a generic infrastructure to
  - **Support virtual game communities.**
  - **Register new games**
  - **Register new players**
  - **Organize tournaments**
  - **Keeping track of the players scores.**
- Provide a framework for tournament organizers
  - **to customize the number and sequence of matchers and the accumulation of expert rating points.**
- Provide a framework for game developers
  - **for developing new games, or for adapting existing games into the ARENA framework.**
- Provide an infrastructure for advertisers.

Types of Requirements

- Functional requirements
  - **Describe the interactions between the system and its environment independent from the implementation**
    - **“An ARENA operator must be able to define a new game. “**
- Nonfunctional requirements
  - **Aspects not directly related to functional behavior.**
    - **“The response time must be less than 1 second”**
    - **“The ARENA server must be available 24 hours a day”**
- Constraints (“Pseudo requirements”)
  - **Imposed by the client or the environment**
    - **“The implementation language must be Java “**
    - **ARENA must be able to dynamically interface to existing games provided by other developers**

What should not be in the Requirements?

- System structure, implementation technology
- Development methodology
- Development environment
- Implementation language
- Reusability

- It is desirable that none of these above are constrained by the client.

Requirements Validation

Requirements validation is a quality assurance step, usually performed after requirements elicitation or after analysis. Also at the delivery (client acceptance test)

- Correctness:
  - **The requirements represent the client's view**
- Completeness:
  - **All possible scenarios, in which the system can be used, are described**
- Consistency:
  - **There are no requirements that contradict each other.**

Requirements Validation

- Clarity:
  - **Requirements can only be interpreted in one way**
- Realism:
  - **Requirements can be implemented and delivered**
- Traceability:
  - **Each system behavior can be traced to a set of functional requirements**

- Problems with requirements validation:
  - **Requirements change quickly during requirements elicitation**
  - **Inconsistencies are easily added with each change**
  - **Tool support is needed!**

Requirements Evolution

- Tool support for managing requirements:
  - **Store the requirements in a shared repository**
  - **Provide multi-user access to the requirements**
  - **Automatically create a specification document from the requirements**
  - **Allow change management of the requirements**
  - **Provide traceability of the requirements throughout the artifacts of the system.**

- RequisitPro from Rational
  - **http://www.rational.com/products/reqpro/docs/datasheet.html**

Prioritizing requirements

- High priority
  - **Addressed during analysis, design, and implementation**
  - **A high-priority feature must be demonstrated**
- Medium priority
  - **Addressed during analysis and design**
  - **Usually demonstrated in the second iteration**
- Low priority
  - **Addressed only during analysis**
  - **Illustrates how the system is going to be used in the future with not yet available technology**

Different Types of Requirements Elicitation

- Greenfield Engineering
  - **Development starts from scratch, no prior system exists, requirements come from end users and clients**
  - **Triggered by user needs**
- Re-engineering
  - **Re-design and/or re-implementation of an existing system using newer technology**
  - **Triggered by technology enabler**
- Interface Engineering
  - **Projects do not change the underlying functionality of an existing system, but improve its interface, either between the system and humans or between the system and some other (possibly third-party) system(s)**
  - **Triggered by technology enabler or new market needs**

Requirements Elicitation

Tecnica	Serve per	Vantaggi	Svantaggi
Questionari	Rispondere a domande specifiche.	Si possono raggiungere molte persone con poco sforzo.	Vanno progettati con grande accuratezza, in caso contrario le risposte potrebbero risultare poco informative. Il tasso di risposta può essere basso.
Interviste individuali	Esplorare determinati aspetti del problema e determinati punti di vista.	L'intervistatore può controllare il corso dell'intervista, orientandola verso quei temi sui quali l'intervistato è in grado di fornire i contributi più utili.	Richiedono molto tempo. Gli intervistati potrebbero evitare di esprimersi con franchezza su alcuni aspetti delicati.
Focus group	Mettere a fuoco un determinato argomento, sul quale possono esserci diversi punti di vista.	Fanno emergere le aree di consenso e di conflitto. Possono far emergere soluzioni condivise dal gruppo.	La loro conduzione richiede esperienza. Possono emergere figure dominanti che monopolizzano la discussione.
Osservazioni sul campo	Comprendere il contesto delle attività dell'utente.	Permettono di ottenere una consapevolezza sull'uso reale del prodotto che le altre tecniche non danno.	Possono essere difficili da effettuare e richiedere molto tempo e risorse.
Suggerimenti spontanei degli utenti	Individuare specifiche necessità di miglioramento di un prodotto.	Hanno bassi costi di raccolta. Possono essere molto specifici.	Hanno normalmente carattere episodico.
Analisi della concorrenza e delle best practices	Individuare le soluzioni migliori adottate nel settore di interesse	Evitare di "reinventare la ruota" e ottenere vantaggio competitivo	L'analisi di solito è costosa (tempo e risorse)

Requirements Elicitation

- Very challenging activity
- Requires collaboration of people with different backgrounds
  - **Users with application domain knowledge**
  - **Developer with solution domain knowledge (design knowledge, implementation knowledge)**
- Bridging the gap between user and developer
  - ***Scenarios:*** **Example of the use of the system in terms of a series of interactions between the user and the system**
  - ***Use cases:*** **Abstraction that describes a class of scenarios**

Why Scenarios and Use Cases?

- Comprehensible by the user
  - **Use cases model a system from the users' point of view (functional requirements)**
    - **Define every possible event flow through the system**
    - **Description of interaction between objects**
- Great tools to manage a project. Use cases can form basis for whole development process
  - **User manual**
  - **System design and object design**
  - **Implementation**
  - **Test specification**
  - **Client acceptance test**
- An excellent basis for incremental & iterative development
- Use cases have also been proposed for business process reengineering (Ivar Jacobson)

Scenarios

- Scenario: “A narrative description of what people do and experience as they try to make use of computer systems and applications” [M. Carroll, Scenario-Based Design, Wiley, 1995]
- A concrete, focused, informal description of a single feature of the system used by a single actor.
- Scenarios can have many different uses during the software lifecycle

Types of Scenarios

- As-is scenario:
  - **Describes a current situation. Usually used in re-engineering projects. The user describes the system**
    - **Example: Description of Letter-Chess**
- Visionary scenario:
  - **Describes a future system. Usually used in greenfield engineering and reengineering projects**
  - **Can often not be done by the user or developer alone**
    - **Example: Description of an interactive internet-based Tic Tac Toe game tournament**
    - **Example: Description - in the year 1954 - of the Home Computer of the Future.**

Types of Scenarios

- Evaluation scenario:
  - **Description of a user task against which the system is to be evaluated.**
    - **Example: Four users (two novice, two experts) play in a TicTac Toe tournament in ARENA.**
- Training scenario:
  - **A description of the step by step instructions that guide a novice user through a system**
    - **Example: How to play Tic Tac Toe in the ARENA Game Framework.**

How do we find scenarios?

- Don't expect the client to be verbal if the system does not exist
  - **Client understands problem domain, not the solution domain.**
- Don't wait for information even if the system exists
  - **"What is obvious does not need to be said"**
- Engage in a dialectic approach
  - **You help the client to formulate the requirements**
  - **The client helps you to understand the requirements**
  - **The requirements evolve while the scenarios are being developed**

Heuristics for finding scenarios

- Ask yourself or the client the following questions:
  - **What are the primary tasks that the system needs to perform?**
  - **What data will the actor create, store, change, remove or add in the system?**
  - **What external changes does the system need to know about?**
  - **What changes or events will the actor of the system need to be informed about?**
- However, don't rely on questions *and* questionnaires alone
- Insist on task observation if the system already exists (interface engineering or reengineering)
  - **Ask to speak to the end user, not just to the client**
  - **Expect resistance and try to overcome it.**

Example: Accident Management System

- What needs to be done to report a “Cat in a Tree” incident?
- What do you need to do if a person reports “Warehouse on Fire?”
- Who is involved in reporting an incident?
- What does the system do if no police cars are available? If the police car has an accident on the way to the “cat in a tree” incident?
- What do you need to do if the “Cat in the Tree” turns into a “Grandma has fallen from the Ladder”?
- Can the system cope with a simultaneous incident report “Warehouse on Fire?”

Scenario example: Warehouse on Fire

- Bob, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, Alice, reports the emergency from her car.
- Alice enters the address of the building into her wearable computer, a brief description of its location (i.e., north west corner), and an emergency level.
- She confirms her input and waits for an acknowledgment.
- John, the dispatcher, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and sends the estimated arrival time (ETA) to Alice.
- Alice received the acknowledgment and the ETA.

Observations about Warehouse on Fire Scenario

- Concrete scenario
  - **Describes a single instance of reporting a fire incident.**
  - **Does not describe all possible situations in which a fire can be reported.**
- Participating actors
  - **Bob, Alice and John**

Requirements Analysis Document

1. Introduction
   - 1.1 Purpose of the system
   - 1.2 Scope of the system
   - 1.3 Objectives and success criteria of the project
   - 1.4 Definitions, acronyms, and abbreviations
   - 1.5 References
   - 1.6 Overview

2. Current system

3. Proposed system

4. Glossary

3. Proposed system
   - 3.1 Overview
   - 3.2 Functional requirements
   - 3.3 Nonfunctional requirements
     - 3.3.1 Usability
     - 3.3.2 Reliability
     - 3.3.3 Performance
     - 3.3.4 Supportability
     - 3.3.5 Implementation
     - 3.3.6 Interface
     - 3.3.7 Packaging
     - 3.3.8 Legal
   - 3.4 System models
     - 3.4.1 Scenarios
     - 3.4.2 Use case model
     - 3.4.3 *Object model*
     - 3.4.4 *Dynamic model*
     - 3.4.5 User interface-navigational paths and screen mock-ups